1. 软件测试Software testing，测试的目的，测试的类型

1）测试软件是评估其质量、功能和性能的过程。

2）软件测试的主要目标是发现潜在的缺陷和错误，并确保软件在实际使用中的可靠性和稳定性。

3）

功能性：

单元测试：单元测试是对软件的最小可测试单元（通常是函数或方法）进行测试，以确保其按照预期进行操作。

集成测试：集成测试是将多个单元测试组合在一起，测试它们之间的交互和协作，以验证组件之间的正确集成。

系统测试：系统测试是对整个软件系统进行测试，以验证其功能、性能、安全性等方面的要求是否得到满足。

验收测试：验收测试是由最终用户或客户执行的测试，以验证软件是否满足其需求和期望。

非功能性：

与变更相关的测试类型：

1. 单元测试Unit testing，基本原则和使用的方法。
2. 模块化测试（Unit Testing）是软件测试的一种方法，用于测试软件中最小的可测试单元（通常是函数、方法或类）
3. 单一责任原则：每个测试用例应该只测试一个特定的功能或行为。这有助于保持测试用例的简单性和可维护性，并使故障排除更容易。

独立性原则：每个测试用例应该是相互独立的，不依赖于其他测试用例的执行顺序或状态。这样可以确保测试结果的可靠性，并使得在出现问题时更容易定位和修复缺陷。

模拟和隔离原则：在模块化测试中，可能需要模拟或隔离与被测试单元相关的外部依赖，例如数据库、网络请求或其他外部系统。通过使用模拟对象或测试替身，可以更好地控制测试环境，并减少测试的复杂性和依赖性。

自动化原则：模块化测试通常通过编写自动化测试脚本来实现。自动化测试可以提高测试效率，减少人为错误，并支持持续集成和持续交付的实践。

1. JUnit, 核心API

JUnit是一个Java语言的单元测试框架

//JUnit是一个用于Java编程语言的开源测试框架，用于编写和运行单元测试。它提供了一组注解和断言方法，以及测试运行器（test runner）来帮助开发人员编写可重复、自动化和可维护的单元测试。

org.junit.Assert：这个类提供了一系列用于判断和测试结果的静态方法。例如，assertEquals()用于比较两个值是否相等，assertTrue()用于验证一个条件是否为真，assertNotNull ()用于验证一个对象是否为非空间等。

org.junit.Test：这个注解用于标记一个方法作为测试用例。JUnit将通过反射来识别这些带有@Test注解的方法，并执行他们。

org.junit.Before和org.junit.After每个测试方法之前运行的方法，可以用初始测试数据或@After注解表示在每个测试方法中后运行的方法，可以使用来自清理资源或者修复环境。

org.junit.BeforeClass和org.junit.AfterClass：这些注解用于标记在整个测试类之前和之后执行的方法。@BeforeClass注解表示在执行测试类中的任何测试方法之前运行的方法，通常用于执行@AfterClass注解表示在执行测试类中的所有测试方法之后运行的方法，通常用于执行一次性的清理工作。

org.junit.runner.JUnitCore：这个类是JUnit的运行器，用于执行测试类或测试套件。可以使用它的静态方法runClasses()来运行一个或多个测试类，并获得测试结果。

1. 自动化构建系统，目的、操作原理、系统示例。

自动化构建系统是一种用于自动化软件开发中构建和部署过程的工具。它们旨在简化和加速软件交付的过程，确保代码从编写到部署的无缝集成和持续交付。

1. Make

make是一种常用的构建自动化工具，通常在Unix和类Unix系统上使用。它使用名为"makefile"的文本文件来定义构建过程中的目标和规则。

makefile包含了一系列的规则，每个规则定义了目标（target）和构建规则（build rule）。每个目标都代表一个要构建的文件或一个要执行的操作。构建规则则定义了如何生成目标文件的方法。

一个makefile中的规则通常包括以下几个部分：

目标（Target）：目标是makefile中的主要组成部分，代表一个要构建的文件或执行的操作。目标可以是文件名、伪目标（不代表实际文件）或操作名称。

依赖（Dependencies）：依赖是目标所依赖的文件或其他目标。当目标的依赖发生变化时，make会检查依赖项的状态并相应地更新目标。

命令（Commands）：命令是构建目标的具体步骤。当目标的依赖项发生变化时，命令将被执行以生成或更新目标文件

通过运行make命令，make会读取makefile文件并根据目标和规则来执行构建过程。它会自动检测依赖项的变化，并根据需要重新构建相关的目标。

1. Ant

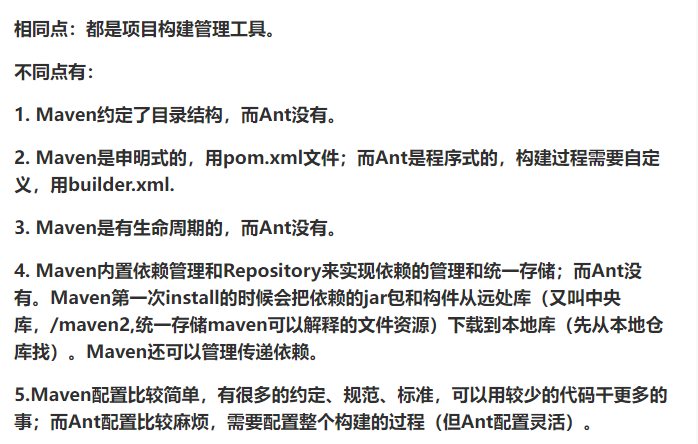
Ant是一种基于Java的构建工具，用于自动化构建和部署Java项目。它使用XML文件作为构建脚本，其中定义了构建过程中的目标、任务和依赖关系。

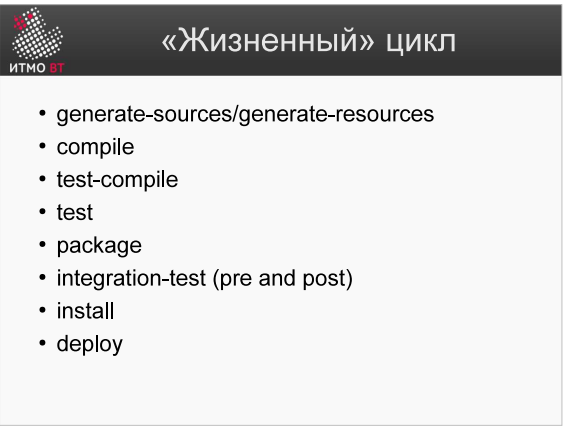
Ant的构建脚本称为构建文件（build file），通常命名为"build.xml"。构建文件由一系列目标（target）和任务（task）组成。

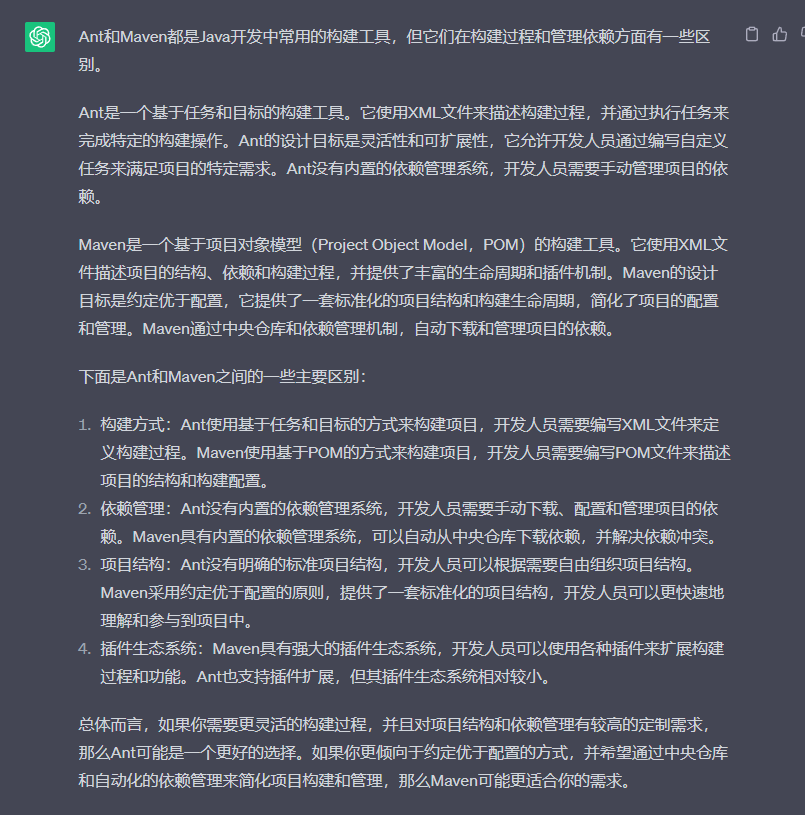
目标是构建过程中的一个步骤或输出，可以是编译源代码、打包文件、运行测试等。每个目标由一个唯一的名称标识。

任务是执行构建过程中的具体操作。任务可以是编译Java代码、运行脚本、复制文件等。Ant提供了许多内置任务，也可以通过自定义任务来扩展功能。

构建文件中的目标可以有依赖关系，即一个目标可能依赖于其他目标的执行结果。当执行一个目标时，Ant会自动检查并按照依赖关系的顺序执行所需的目标。







declarative是声明式

imperative是命令式

